005140.P5252

UNITED STATES PATENT APPLICATION
FOR
DISCOVERY AND CONFIGURATION OF NETWORK ATTACHED STORAGE DEVICES

INVENTORS:

STEPHEN D. PAUL
a citizen of The United States, residing at
820 EAST HEARTSTRONG STREET
SUPERIOR, CO 80027

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026
(303) 740-1980

DISCOVERY AND CONFIGURATION OF NETWORK ATTACHED STORAGE DEVICES

FIELD OF THE INVENTION

[0001] The invention relates generally to the field of computer networks. More particularly, the invention relates to dynamic discovery and configuration of a device added to a network.

BACKGROUND OF THE INVENTION

[0002] Various methods of maintaining computer networks exist. Sun Microsystems' Jini™ network technology is an example of one such method Jini is a trademark of Sun Microsystems Inc. of Palo Alto, California. Jini network technology provides simple mechanisms that enable devices to plug together to form a network. Each device provides services that other devices in the network may use. These devices provide their own interfaces, which ensures reliability and compatibility.

[0003] Jini technology uses a lookup service with which devices and services register. When a device plugs in, it goes through an add-in protocol, called discovery and join-in. The device first locates the lookup service (discovery) and then uploads an object that implements all of its services' interfaces (join).

[0004] To use a service, a person or a program locates it using the lookup service, a process referred to as lookup. The service's object is copied from the lookup service to the

requesting device where it will be used. The lookup service acts as an intermediary to connect a client looking for a service with that service. Once the connection is made, the lookup service is not involved in any of the resulting interactions between that client and that service.

[0005] As mentioned above, discovery and join are the processes for adding a service to a system using a lookup service such as Jini. Figure 1 is a block diagram illustrating standard discovery, registration or join, look-up, and service invocation processes for a system utilizing a lookup service. A service provider 110 is the originator of the service such as a device or software. First, the service provider 110 locates a lookup service 100 by multicasting a request 125. The lookup service 100 responds to the request 125, thereby identifying itself to the service provider 110.

[0006] After a lookup service 100 has been located, a service object 130 for the service 110 is loaded into the lookup service 100. This service object 130 contains a public interface for the service 110 including the methods that users and applications will invoke to execute the service 110, along with any other descriptive attributes 135. In the Jini network technology, these public interfaces are Java™ programming language based objects, Java is a Trademark of Sun Microsystems Inc. of Palo Alto, California

[0007] A client 105 locates an appropriate service object 130 within a lookup service 100 by searching for the particular type of service. That is, the client 105 searches the lookup service 100 for a particular service 110 identified by its interface or service object 130 written in the Java programming language, along with descriptive service attributes 135 that are used in a user interface. The service object 130 is loaded 120 into the client 105. The final stage is for the client 105 to invoke the service 110.

[0008] The client 105 uses the service object 145 obtained from the lookup service 100 to directly access 140 the service provider 110. Such a method allows a client 105 to directly communicate 140 with a service provider 110 using a service object 145 that provides a public interface written specifically for the service provider 110 thereby providing increased scalability and compatability.

[0009] However, a few problems exist with such a method. Currently, when adding a new device such as a network attached storage device to the network, the port or IP address of the device must be known in order to configure the device. That is, there is no way to dynamically discover the addition or removal of a device such as a network attached storage device. Additionally, once discovered, there is no way to configure it from a remote site.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The appended claims set forth the features of the invention with particularity. The invention, together with its advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

[0011] **Figure 1** is a block diagram illustrating standard discovery, registration or join, look-up, and service invocation processes for a system utilizing a lookup service;

[0012] **Figure 2** is a block diagram illustrating an exemplary network upon which the present invention may be implemented;

[0013] **Figure 3** is a block diagram illustrating a storage device according to one embodiment of the present invention;

[0014] **Figure 4** is a flowchart illustrating a conceptual view of one embodiment of the present invention;

[0015] **Figure 5** is a flowchart illustrating processing on a storage device according to one embodiment of the present invention; and

[0016] **Figure 6** is a flowchart illustrating processing on an administration terminal according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0017] According to one embodiment of the present invention, software resides on the NAS device itself to automatically look for a Jini lookup service within a specific NAS device domain. Once found, the NAS device registers itself as a NAS device with the Jini lookup service. This allows any client software looking for a registration of this type to dynamically see the addition of the storage device. If the NAS device is removed from the network, the NAS service Jini; lease will expire and all interested parties will be notified of the removal, thus providing the dynamic NAS device removal capability.

[0018] Additionally, software resides on the storage device providing device specific instructions to implement a set of generic commands for interfacing with the device. A generic Application Programming Interface (API) is provided which all equipped storage devices will implement. This API is device independent. However, the implementation of it on the storage device itself is dependent on the internals of the device. This removes the burden of device specific knowledge from the user. With this implemented the user can configure multiple and diverse storage devices from a generic Graphical User Interface (GUI) regardless of the types of devices used. All device specific processing is performed by the devices themselves.

[0019] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

[0020] The present invention includes various steps, which will be described below. The steps of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware and software.

[0021] The present invention may be provided as a computer program product which may include a machine-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0022] Importantly, while embodiments of the present invention will be described with reference to Sun Microsystems Jini network technology, the method and apparatus described herein are equally applicable to other network technologies that provide a network based directory look-up service.

[0023] Figure 2 is a block diagram illustrating an exemplary network upon which the present invention may be implemented. In this example an Ethernet network 210 is shown. Such a network may utilize the common Transmission Control Protocol/Internet Protocol (TCP/IP). Of

course, many other types of networks and protocols are available and are commonly used. However, for illustrative purposes, Ethernet and TCP/IP will be referred to.

[0024] Connected to this network 210 are computers 220, 230, and 260 and other devices 240 and 250. The computers 220, 230, and 260 may include a network administrator terminal 220. A human operator could use this network administrator terminal 220 to monitor and maintain the network. The devices 240 and 250 may include network attached storage devices or other types of non-terminal devices. The number and arrangement of this equipment may vary depending on the application.

[0025] According to one embodiment of the present invention, software resides on the NAS device itself to automatically look for a Jini lookup service within a specific NAS device domain. Once found, the NAS device registers itself as a NAS device with the Jini lookup service. This allows any client software looking for a registration of this type to dynamically see the addition of the storage device. If the NAS device is removed from the network, the NAS service Jini lease will expire and all interested parties will be notified of the removal, thus providing the dynamic NAS device removal capability.

[0026] Additionally, software resides on the storage device providing device specific instructions to implement a set of generic commands for interfacing with the device. A generic configuration API is provided which all equipped NAS devices will implement as part of the Jini service. This API is device independent. However, the implementation of it on the NAS device itself is dependent on the internals of the device. This removes the burden of device specific knowledge from the user. With this implemented the user can configure multiple and diverse NAS devices from a generic GUI regardless of the types of devices used. All device specific processing is performed by the devices.

[0027] Figure 3 is a block diagram illustrating a storage device according to one embodiment of the present invention. The storage device 300 is preloaded with interface software 310. This interface software 310 is written specific to the hardware of the device 300 and implements a set of generic commands for interfacing with the device 300. The standard Jini service object 305 received from a look up service can then be used by other devices on the network to access the interface 310 on the storage device 300.

[0028] The interface 310 is a set of functions that other devices can use. These functions are very generic and may include functions such as create disk, create file system, delete disk, delete file system, add share, etc. According to one embodiment of the present invention, these functions consist of Java code implemented for a specific device. Advantageously, the Application Program Interface (API) calls to the interface are the same for all devices, regardless of type. Details of processing are hidden from user and performed automatically on the storage device. Therefore, the user need not be concerned about the type of device being used.

[0029] Figure 4 is a flowchart illustrating a conceptual view of one embodiment of the present invention. Initially, at processing block 400, the storage device is started. This process typically includes starting the operating system of the device and performing all required configuration. Next, at processing block 405, the storage device announces its presence on the network by writing its service object to the lookup service. At processing block 410, the administration terminal reads the lookup service and if a new device is detected at decision block 415, the administration terminal informs a human operator of the presence of the new device through a graphical user interface at processing block 420. If no new device is detected at processing block 415, the administration terminal continues to periodically read the lookup service at processing block 410. Finally, at processing block 430, the administration terminal

uses the service object from the lookup service to send API calls to the interface software on the storage device in response to commands from the human operator using the graphical user interface, for example.

[0030] Figure 5 is a flowchart illustrating processing on a storage device according to one embodiment of the present invention. Initially, once the storage device is connected to the network and powered up, the operating system for the device is started at processing block 505. This process includes performing all necessary setup for using the lookup service. Next, at processing block 510, the device looks for a lookup service on the network. Once a lookup service is located, the storage device registers with the lookup service at processing block 515 by writing its service object and service attributes to the lookup service. Finally, at processing block 525, the storage device receives API calls from an administration terminal and, using the preloaded interface, executes the device specific commands to perform the system calls.

[0031] The interface is a set of functions that other devices can use. These functions are very generic and may include functions such as create disk, create file system, delete disk, delete file system, add share, etc. According to one embodiment of the present invention, these functions consist of Java code implemented on a specific device. Advantageously, the interface is the same for all devices. That is, the API calls are the same for all devices, regardless of type. Details of processing are hidden from user and performed automatically on the storage device. Therefore, the user need not be concerned about the type of device being used.

[0032] Figure 6 is a flowchart illustrating processing on an administration terminal according to one embodiment of the present invention. Initially, once the administration terminal is connected to the network and powered up, the operating system for the terminal is started at processing block 605. This process includes performing all necessary setup for using the lookup

service. Next, at processing block 610, the administration terminal looks for a lookup service on the network. Once a lookup service is located, the administration terminal, at processing block 615, looks for a storage device with a particular signature.

[0033] If a device is not found at decision block 620, the administration terminal periodically rechecks the lookup service at processing block 615. If a device is found at decision block 620, the administration terminal reads the service object from the lookup service at processing block 625 and displays a notification of the presence of a new device to a human operator via a graphical user interface at processing block 630. The administration terminal then waits for an event from the graphical user interface at decision block 645. Such an event may be a human operator clicking a button on the graphical user interface to execute a command on the storage device. Once an event is detected at decision block 645, the administration terminal sends the proper sequence of API calls to the storage device at processing block 650.

[0034] According to one embodiment of the present invention, the administration terminal opens a GUI and displays information about the network in tree form with panels and buttons to administrator devices. This GUI provides a graphical interface to allow the use of the public interface functions of the device. The GUI opens panels to display information about the network and allow administration of selected disks. By using this interface, a human operator can configure a number of devices, potentially of different types. The GUI takes button clicks and converts them into a series of function calls to the storage device in the proper sequence so that the user need not be concerned with the lower level commands. So, with simple button clicks, an administrator can create a disk and file system and share files with all users on a network without regard to the type of device being used.